SYSTEM AND METHOD FOR SWITCHING VARIABLY SIZED INFORMATION GROUPS

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to the switching of information packets and, more particularly, to a system and method for arbitrating between switch inputs and switch outputs.

2. Description of the Related Art

As noted in US Patent 6,285,679 (Dally et al.), data communication between computer systems for applications such as web browsing, electronic mail, file transfer, and electronic commerce is often performed using a family of protocols known as IP (internet protocol) or sometimes TCP/IP. As applications that use extensive data communication become more popular, the traffic demands on the backbone IP network are increasing exponentially. It is expected that IP routers with several hundred ports operating with aggregate bandwidth of Terabits per second will be needed over the next few years to sustain growth in backbone demand.

The network is made up of links and routers. In the network backbone, the links are usually fiber optic communication channels operating using the SONET (synchronous optical network) protocol. SONET links operate at a variety of data rates currently ranging from OC-3 (155 Mb/s) to OC-192 (9.9 Gb/s). These links, sometimes called trunks, move data from one point to another, often over considerable distances.

Routers connect a group of links together and perform two functions: forwarding and routing. A data packet arriving on one link of a router is forwarded by sending it out on a different link depending on its eventual destination and the state of the output links. To compute the output link for a given packet, the router participates in a routing protocol where all of the routers on the Internet exchange information about the connectivity of the network and compute routing tables based on this information.

Most prior art Internet routers are based on a common bus or a crossbar switch. In the bus-based switch of a SONET link, a line-interface module extracts the packets from the incoming SONET stream. For each incoming packet, the line interface reads the packet header, and using this information, determines the output port (or ports) to which the packet is to be forwarded. To forward the packet, the line interface module arbitrates for the common bus. When the bus is granted, the packet is transmitted over the bus to the output line interface module. The module subsequently transmits the packet on an outgoing SONET link to the next hop on the route to its destination.

Bus-based routers have limited bandwidth and scalability. The central bus becomes a bottleneck through which all traffic must flow. A very fast bus, for example, operates a 128-bit wide datapath at 50 MHz giving an aggregate bandwidth of 6.4 Gb/s, far short of the Terabits per second needed by a backbone switch. Also, the fan-out limitations of the bus interfaces limit the number of ports on a bus-based switch to typically no more than 32.

The bandwidth limitation of a bus may be overcome by using a crossbar switch. For N line interfaces, the switch contains N(N-1) crosspoints. Each line interface can select any of the other line interfaces as its input by connecting the two lines that meet at the appropriate crosspoint. To forward a packet with this organization, a line interface arbitrates for the required output line interface. When the request is granted, the appropriate crosspoint is closed and data is transmitted from the input module to the output module. Because the crossbar can simultaneously connect many inputs to many outputs, this organization provides many times the bandwidth of a bus-based switch.

Despite their increased bandwidth, crossbar-based routers still lack the scalability and bandwidth needed for an IP backbone router. The fan-out and fan-in required by the crossbar connection, where every input is connected to every output, limits the number of ports to typically no more than 32. This limited scalability also results in limited bandwidth. For example, a state-of-the-art crossbar might operate 32 different 32-bit channels simultaneously at 200 MHz giving a peak bandwidth of 200 Gb/s. This is still short of the bandwidth demanded by a backbone IP router.

Fig. 1 is a schematic block diagram illustrating a conventional packet switch (prior art). As noted in US Patent 6,275,491 (Prasad et al.), the architecture of conventional fast packet switches may be considered, at a high level, as a number of inter-communicating processing blocks. In this switch, ports P_0 through P_n are in communication with various nodes, which may be computers or other switches (not shown). Each of ports receive data over an incoming link, and transmits data over an outgoing link. Each of ports are coupled to switch fabric F, which effects the routing of a message from the one of input ports, to the one of *n* output ports associated with the downstream node on the path to the destination of the packet. The switch has sufficient capability to divide the packet into slices (when on the input end) and to reconstruct slices into a packet (when on the output end). Arbitor A is provided to control the queuing of packets into and out of switch fabric F, and to control the routing operation of switch fabric F accordingly.

While the high-level architecture of fast packet switches may be substantially common, different architectural approaches are used in the implementation of the fast packet switch. These approaches determine the location (input, output, or both) and depth of cell queues or buffers, and also the type of routing used within switch fabric. For example, one architecture may operate by the input ports forwarding each received cell immediately to switch fabric F, which transfers cells at its input interfaces to its output interfaces in a time-division multiplexed fashion; on the output side, each cell that is output from switch fabric F is appended to a FIFO queue at its addressed output port. Another architecture may utilize input queues at the input ports, with arbitor A controlling the order in which cells are applied from the input queues to switch fabric F, which operates in a crossbar mode. Another architecture may utilize both input and output queues at the input ports, with switch fabric F and arbitor A operating as a multistage interconnection network. These and other various architectures are known in the field of fast packet switching.

Also as is well known in the art, actual communication traffic is neither uniform nor independent; instead, real traffic is relatively bursty, particularly in the communication of data and compressed video. As such, traffic management algorithms are often utilized in fast packet switching to manage the operation of the switch and to optimize switch performance. Examples of well-known traffic management algorithms include traffic shaping, flow control, and scheduling.

As noted in US Patent 6,073,199 (Cohen et al.), arbitors are used in computer systems to control access to a common bus used by multiple devices. Arbitors typically use arbitration schemes such as fixed priority, round robin, or rotating priority. A fixed priority algorithm assigns a priority to each device on the bus and grants usage based upon the relative priority of the devices making the requests. The round robin scheme has a fixed order and grants bus usage based upon the requestor order and the current user of the bus. The rotating priority scheme changes the priority of requestors based on a fixed algorithm. A deficit round robin algorithm is essentially the combination of the round robin algorithm with a system that gives an advantage or "credit" to an entity denied a grant. Conventionally, the fairness inherent in the DRR process is offset by the sequential steps required for implementation.

The goal of all arbitration schemes is to insure fair access to the shared resource, and to efficiently grant the resource to the correct requestor. The fixed priority scheme is unfair because a high priority requestor can consume all the shared resource, starving the lower priority requestors. The round robin scheme is inefficient because multiple clocks may be required to determine which requestor should be granted the resource. Also round robin schemes have a fixed grant pattern that can result in starvation of particular requestors if request patterns match the round robin grant pattern. Rotating priority schemes are random in their efficiency and fairness based on the algorithm chosen to update device priority.

It would be advantageous if a scheduling algorithm could be devised for the efficient transfer of information packets having an unspecified length, or number of cells.

It would be advantageous if variable length information packets could be scheduled for transfer across a switch with a minimum of overhead devoted to the scheduling decision process.

SUMMARY OF THE INVENTION

Conventional switches process information packets having a fixed length, or fixed number of cells. In response to the fixed length information packets, these switches necessarily make new switching decisions after the processing of each information packet. This rigid decision making approach is not always efficient. The present invention switching algorithm is able to process information packets of variable size. To accommodate these variable length information packets, the present invention algorithm locks the switch output/switch input links until the information is completely transferred. As a result, more information can be transferred with a reduced amount of processing overhead.

Accordingly, a method is provided for transferring information across a switch connection. The method comprises: accepting information packets, having a variable number of cells, at a plurality of switch inputs; parsing the information packets into lengths of a cell;

linking switch inputs to switch outputs; and, locking the links to transfer the information packets in units of one cell per decision cycle, until the transfer is complete.

Typically, the information packets are received at a number of switch inputs, all addressed to a common (the same) switch output. Then, the method further comprises: for each common switch output, arbitrating between the plurality of available switch inputs (switch inputs having information packets addressing that particular switch output); and, linking switch inputs to switch outputs in response to arbitration.

Arbitrating between the available switch inputs includes establishing an available input priority list for each switch output, to select the least recently used available switch input. The process is enabled by arbitrating between the available switch inputs in a plurality of arbitration cycles. It is a feature of the invention that each arbitrating switch output simultaneously nominates a switch input selection.

Once the switch inputs are nominated, another process of arbitration occurs for each switch input that has received multiple nominations. As with switch input nomination process, the least recently used nominating switch output is accepted, enabled with a nominating output priority list. Again, the arbitrating switch inputs simultaneously accept nominating switch outputs. The method promotes fair distribution and efficient communication with a minimum of overhead processing time.

Details of the above-mentioned method for transferring information across a switch connection, as well as an arbitration system for transferring information across a switch, are provided below.

BRIEF DESCRIPTION OF THE DRAWING

Fig. 1 is a schematic block diagram illustrating a conventional packet switch (prior art).

Figs. 2a and 2b are schematic block diagrams illustrating a system for transferring information across a switch.

Fig. 3 illustrates an available input priority list, such as might be used by the arbiter of Fig. 2a.

Fig. 4 illustrates a nominating output priority list, such as might be used by the arbiter of Fig. 2a.

Fig. 5 is a schematic block diagram illustrating the links formed in the example presented in Figs. 3 and 4.

Fig. 6 is a flowchart illustrating a method for transferring information across a locked switch connection.

Fig. 7 is a flowchart depicting in greater detail the method of Fig. 6.

Fig. 8 is a flowchart illustrating a method for transferring information across a locked switch connection with a first plurality of switch outputs.

Fig. 9 is a flowchart illustrating additional details of the method of Fig. 8.

Fig. 10 is a flowchart depicting a method for transferring information across a switch connection.

Fig. 11 is a flowchart presenting the arbitration process of Figs. 6 through 10 in a different light.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Some portions of the detailed descriptions that follow are presented in terms of procedures, steps, logic blocks, codes, processing, and other symbolic representations of operations on data bits within a device. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. A procedure, microprocessor executed step, data item, application, logic block, process, etc., is here, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a switch. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, data items, numbers, or the like. Where physical devices, such as a memory are mentioned, they are connected to other physical devices through a bus or other electrical connection. These physical devices can be considered to interact with logical processes or applications and, therefore, are "connected" to logical operations. For example, a memory can store or access code to further a logical operation, or an application can call a code section from memory for execution. Further, a software application can perform functions such as arbitrating and switching.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the present invention, discussions utilizing terms such as "processing" or "connecting" or "determining" or "comparing" or "addressing" or "retrieving" or the like, refer to the action and operations of in a system that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the switch, or switch peripherals.

Figs. 2a and 2b are schematic block diagrams illustrating a system for transferring information across a switch. In Fig. 2a, the system 100 comprises a queuing device 102 having a plurality of inputs to accept information packets with a variable number of cells. Specifically, a first input is connected to line 104, a second input to line 106, and an *n*th input to line 108. The queuing device 102 is not limited to any particular number of inputs, there can be any number of inputs between the second and *n*th inputs. The queuing device 102 accepts information concerning the length of information packets on line 109. The length information is eventually used to lock switch links to complete the transfer of entire information packets without intervening arbitration processes. In some aspects of the invention the queuing device determines the information packets lengths in the event that the received length data is incorrect. The queuing device 102 has a plurality of outputs to supply the information packets. Outputs are shown connected to lines 110, 112, and 114.

The queuing device essentially passes through the information packets accepted on lines 104, 106, and 108 to a switch, after some information packet differentiation. As discussed in more detail below, the information packets are queued into queues differentiated by information packet switch output address. Further, in some aspects of the invention each information packet has a graded rank or class of service (COS), and the information packets are differentiated by channel. Thus, each queuing device output differentiates the information packets by channels and queues for presentation to the switch input. However, for simplicity each output is depicted as a single line. In some aspects of the invention a priority queue unit (not shown) is interposed between the queuing device 102 and the switch to select the channels to be presented to the switch input.

The queuing device has a control output on line 115 to supply status messages that communicate the length of the accepted information packets, as well as some content information, such as COS and destination, which is defined herein as status information. As is presented below, this status information is used in the arbitration process that connects switch inputs to switch outputs.

A switch 116 has a plurality of inputs connected to the queuing device outputs on lines 110, 112, and 114 and a plurality of switch outputs on lines 118, 120, and 122. As shown, switch 116 has a first (line 110), second (line 112), and *n*th (line 114) input, but there may be any number of inputs between the second and *n*th inputs. Likewise the switch 116 is shown with a first (line 118), second (line 120), and *n*th output (line 122). Although the switch 116 is not limited to any particular number of inputs or outputs, it is typical, but not necessary, that the number of switch inputs equal the number of switch outputs. The switch 116 connects switch inputs to switch outputs in response to commands at a control input of the switch on line 124. The switch 116 locks the link (connection) between a selected switch input and a switch output to transfer the information packets.

For simplicity only two switch sections are shown. A first switch section 116a is shown connected to the switch input on line 110. As shown, line 110 can be connected to lines 118, 120, or line 122. That is, line 110, as well as all the other switch inputs, can be connected to any of the switch outputs. A second switch section 116*n* is shown connected to line 114. Again, line 114 can be connected to any of the switch outputs.

The queuing device 102 processes the information packets in units of one cell per decision cycle, where a cell is defined to be a predetermined number of bytes. The present invention is not limited to any particular length or definition of cell. The switch 116 transfers the contents of each information packet in units of a cell.

For example, the queuing device 102 accepts a first information packet at a first input on line 104, with a first plurality of cells, addressed to a first switch output on line 118. The queuing device 102 supplies the first information packet at a first queuing device output on line 110. The switch 116 has a first input on line 110 connected to the first output of the queuing device 102. The switch 116 links the first input on line 110 to the first switch output on line 118 in response to switching commands on line 124. The switch 116 transfers the contents of the first information packet, from the first switch input to the first switch output, one cell at a time (per decision cycle), until the transfer is complete.

As mentioned above, the queuing device 102 accepts a plurality of information packets at each input addressed to a plurality of switch outputs. That is, the information packets accepted at any particular queuing device input can be addressed to any of the switch outputs. Thus, it is often the case that multiple information packets arrive at the switch 116 on two or more input lines, addressing a common (the same) switch output. It is not desirable to connect two switch input lines to the same output. In these situations the system 100 requires arbitration to select a switch input for linking to a switch output.

An arbiter 130 has an input on line 115 to receive the status message from the queuing device 102 and an output connected to the control input of the switch on line 124 to supply switch commands. The arbiter 130 arbitrates between the plurality of available switch inputs addressing the common switch output. An available switch input is defined in context of a particular switch output, where an available switch input is an input that includes an information packet addressing that particular switch output. The arbiter 130 supplies switch commands to the control input of the switch 116 in response to arbitrating the links. Then, the switch 116 links the selected switch input to the switch output in response to switch commands from the arbiter 130.

There are a variety of possible schemes that can be used to arbitrate between switch inputs competing for a common switch output.

Preferably, the arbiter 130 arbitrates between the plurality of available switch inputs by selecting the least recently used available input. Thus, the selection of switch inputs is more evenly distributed.

To accomplish the switching, the queuing device 102 arranges the information packets into a first plurality of virtual output queues, in response to the switch output to which the information packets are addressed. For example, input line 104 of queuing device 102 is channeled into a first plurality of queues, including line 104a, with information packets addressed to the first switch output on line 118. Line 104b has information packets addressed to the second switch output on line 120 and line 104*n* has information packets addressed to the *n*th switch output on line 122. Although line 110 is depicted as a single line for simplicity, the virtual output queues are again represented in the switch as lines 110a, 110b, and 110*n*. Likewise, each switch input and queuing device output line includes a first plurality of virtual output queues. Then, the arbiter 130 arbitrates between the plurality of available switch inputs (having virtual queues addressing a common switch output) and supplies switch commands to link a selected virtual output queue, in a selected switch input, to an arbitrating switch output.

Fig. 2b is an alternate representation of the switch 116 of Fig. 2a detailing the switch inputs. Assuming $n = 3$, the first switch input (first queuing device output) on line 110 includes three queues, one for each of the switch outputs. Likewise, the second switch input on line 112 includes three queues, one for each of the switch outputs. Switch input *n* also has three queues. The invention is not limited to any particular number of queues, but typically there is a queue for every switch output.

The queuing function makes it much more likely that a plurality of switch inputs will be vying for a common output. It should also be noted that there need not be information packets in every queue. For example, if at the first switch input there are no information packets addressing the second switch output (line 110b), then that output queue will be empty.

Returning to Fig. 2a, the arbiter 130 nominates the highest priority available switch input for each arbitrating switch output in a first arbitration cycle. If the nominated switch input is not selected, the arbiter nominates successively lower priority available switch inputs in subsequent arbitration cycles. In each arbitration cycle, the arbiter 130 simultaneously nominates an available switch input for each arbitrated switch output.

The arbiter 130 establishes an available input priority list by creating a sequential input pointer for each switch output, and nominates switch inputs in response to the available input priority list. Following the acceptance of a nominating switch output by a first available switch input in a first arbitration cycle, the arbiter 130 sets the pointer to a second switch input next in sequence to the first available switch input. The arbiter 130 nominates the highest priority available input in a subsequent arbitration cycle by nominating the available switch input closest in succession to the second switch input.

Fig. 3 illustrates available input priority lists, such as might be used by the arbiter 130 of Fig. 2a. Such a priority list can be used to implement the least recently used input arbitration scheme. In some aspects of the invention, the arbiter 130 establishes an available input priority list at each output, and selects inputs in response to the available input priority list. Using the priority lists, the arbiter 130 arbitrates between the plurality of available switch inputs addressing the common switch output in a plurality of arbitration cycles.

For example, it is assumed that switch output 1 (the first switch output) has switch input 2 and switch input *n* as available inputs. Alternately stated (referring momentarily to Fig. 2b), the queues of the second switch input (line 112) and the *n*th switch input (line 114) have information packets addressed to the first switch output. Returning to Fig. 3, it is also assumed that switch output 2 (the second switch output) has switch input 1, switch input 2, and switch input *n* as available switch inputs. Switch output *n* is assumed to have inputs 1, 2, and *n* as available switch inputs. In response to arbitration in a previous decision cycle, the pointer of switch output 1 is directed at input 1, the pointer of switch output 2 is directed at input 1, and the pointer of switch output *n* is directed at input 2. In a first arbitration cycle, the available input priority list for switch output 1 list would select input 1 in response to the pointer. However, input 1 is not available (as assumed above). Therefore, the pointer goes to the next input in succession, input 2. Since input 2 is an available switch input, it is nominated. At the available input priority list for switch output 2, the pointer is directed at input 1, an available input, and input 1 is nominated. Since there is no contention for input 1, it is also selected. At the available input priority list for switch output *n*, the pointer is directed at input 2, an available input, and input 2 is nominated. The nomination process for all three outputs occurs simultaneously.

To continue the example, both switch output 1 and switch output *n* have nominated the second input. Returning to Fig. 2a, further arbitration must occur to resolve the occurrence of multiple nominating outputs vying for the same input. Thus, the arbiter 130, following the nomination of switch inputs for arbitrating switch outputs, arbitrates between nominating switch outputs for each switch input receiving multiple nominations. As in the available input arbitration process, the arbiter 130 accepts the least recently used nominating switch outputs in arbitrating between the nominating switch outputs.

The arbiter 130 accepts the least recently used nominating switch outputs by establishing a nominating output priority list for each switch input. The arbiter 130 accepts nominating switch outputs in response to the nominating output priority list. Once again, the arbiter 130 simultaneously accepts a nominating switch output for each arbitrating switch input.

The arbiter 130 establishes a nominating output priority list by creating a sequential output pointer for each switch input. Following the acceptance of a nominating first switch output in a first decision cycle, the arbiter sets the pointer to a second switch output next in sequence to the first switch output. Then, the arbiter 130 accepts the highest priority nominating switch output in a subsequent arbitration cycle by accepting the nominating switch output closest in succession to second switch output.

Fig. 4 illustrates a nominating output priority list, such as might be used by the arbiter 130 of Fig. 2a. In the above example used in the discussion of Fig. 3, a link has been established between switch output

2 and switch input 1. Therefore, no arbitration need be performed for input 1. None of the switch outputs have nominated input *n*, therefore, no arbitration need be performed with respect to input *n*. However, both switch output 1 and switch output *n* have selected input 2. The nominating output priority list pointer for the second input (input 2) is initially set to output 2. Since output 2 is not a nominating switch output, the pointer goes to the next switch output in succession, switch output *n*. Therefore, switch output *n* is accepted, and a link can be established between switch input 2 and switch output *n*. Alternately stated, switch output *n* has selected the second switch input. The first arbitration cycle is ended and the pointer for switch input 2 goes to the next output in succession, output 1.

Returning to Fig. 3, a link has yet to be formed for the first switch output. Initially (in the first arbitration cycle), inputs 2 and *n* were available switch inputs for switch output 1. Input 2 is no longer available, as it is now locked to switch output *n*. Therefore, input *n* is now nominated and selected, since there is no contention for switch input *n*. In the event that multiple switch inputs had been available, the available input priority list pointer for switch output 1 would have been used to select the next available switch input in succession. At the end of the second arbitration cycle, the pointer for the available input priority list for switch output remains directed at input 1, as a switch input link was not created in the first arbitration cycle. The pointer for output list 2 is set to input 2, the next input in succession to selected input 1, and the pointer for output list *n* is set to input *n*, the next input in succession to the selected input 2. The decision cycle ends and a cell is transferred through each of the links. The example presented above uses a two cycle arbitration process, two steps of output arbitration and one step of input arbitration, however, the invention is not limited to any particular number of arbitration cycles. Although the example depicts a situation where all the outputs are linked to an input, it is possible that after two cycles of arbitration that some outputs could remain unlinked. It is a design decision to trade off processing time (additional arbitration cycles) against greater throughput per decision cycle. In some aspects of the invention, the arbitration process concludes with either biased or random mechanisms to link unselected switch inputs to switch outputs, to promote throughput at the expense of fairness.

Fig. 5 is a schematic block diagram illustrating the links formed in the example presented in Figs. 3 and 4.

Returning to Fig. 2a, the system further comprises a timer 140 having an output to supply a decision cycle signal including a plurality of arbitration cycles. The switch 116 has an input connected to the output of the timer 140 on line 142. The switch 116 locks the links to transfer the information packets contents in units of one cell per link, every decision cycle in response to signals from the timer 140. The arbiter 130 arbitrates between switch inputs and switch outputs in response to signals from the timer 140.

In some aspects of the invention, the timer 140 supplies a decision cycle signal with a maximum number of arbitration cycles per decision cycle. If each switch output has not selected a switch input, the arbiter 130 continues arbitration in a subsequent arbitration cycle. The arbiter 130 ends arbitration if each switch output has selected a switch input, or the arbiter 130 ends arbitration if the maximum number of arbitration cycles is reached.

The arbiter 130 sends commands on line 124 to lock the switch links, by checking the number of cells in the information packet remaining to be transferred for each switch output locked to a switch input. The length information is supplied by the queuing device 102. Alternately, the queuing device 102 can perform this function directly. If the number of remaining cells is one or greater, the arbitration process is bypassed for that link in the subsequent decision cycle. However, the arbiter 130 arbitrates between a plurality of switch inputs for a common switch output in the subsequent decision cycle, if no cells remain in the information packet to be transferred. It should also be understood that if there is only one available switch input for a particular switch output, the switch output selects the available switch input without engaging in arbitration.

Fig. 6 is a flowchart illustrating a method for transferring information across a locked switch connection. Although the method (and the methods described below) is depicted as a sequence of numbered steps for clarity, no order should be inferred from the numbering unless explicitly stated. The method begins with Step 600. Step 602 accepts information packets having a variable number of cells, at a plurality of switch inputs, addressed to a plurality of switch outputs. Step 604 parses the information packets into lengths of a cell. Step 606 links switch inputs to switch outputs. Step 608 locks the links to transfer the information packets. Locking the links to transfer the information packets in Step 608 includes transferring the information packet contents in units of a cell per decision cycle.

In some aspects of the invention, accepting information packets having a variable number of cells, at a plurality of inputs in Step 602 includes accepting a first information packet at a first input, with a first plurality of cells, addressed to a first output. Linking switch inputs to switch outputs in Step 606 includes linking the first switch input to the first switch output. Locking the links to transfer the information packets in Step 608 includes transferring the contents of the first information packet, from the first input to the first output, one cell per decision cycle, until the transfer is complete.

Accepting information packets having a variable number of cells, at a plurality of switch inputs, in Step 602 includes a plurality of available switch inputs (as defined above in the description of Fig. 2a) accepting information packets addressed to a common (the same) switch output. Then, Step 602a arbitrates between the available switch inputs for the switch output. Linking switch inputs to switch outputs in Step 604 includes linking the switch input to the switch output in response to arbitration.

Fig. 7 is a flowchart depicting in greater detail the method of Fig. 6. The method begins with Step 700. Step 702 accepts information packets having a variable number of cells, at a plurality of switch inputs, addressed to a plurality of switch outputs. A plurality of available switch inputs accept information packets addressed to a common switch output. Step 704 parses the information packets into lengths of a cell. Step 706 arbitrates between the available switch inputs for the switch output. Step

708 selects the least recently used available switch input. Step 710 links the switch input to the switch output. Step 712, in response to arbitration, locks the link to transfer information packets in units of a cell per decision cycle, until the transfer is complete.

In some aspects of the invention, a first plurality of switch outputs are included. Step 703, following the accepting of the information packets at each switch input, queues the information packets into a first plurality of virtual output queues, differentiated by the switch outputs to which the information packets are addressed. Arbitrating between the plurality of available switch inputs in Step 706 includes arbitrating between a plurality of available switch inputs, where each of the available switch inputs has a virtual output queue with an information packet addressing a common switch output.

In some aspects of the invention, arbitrating between the plurality of available switch inputs in Step 706 includes establishing an available switch input priority list for each switch output. Step 708 selects switch inputs in response to the available switch input priority list.

In some aspects of the invention, arbitrating between the plurality of available switch inputs in Step 706 includes arbitrating in a plurality of arbitration cycles. Arbitrating in a plurality of arbitration cycles in Step 706 includes each arbitrating switch output simultaneously nominating a switch input.

Fig. 8 is a flowchart illustrating a method for transferring information across a locked switch connection with a first plurality of switch outputs. The method begins with Step 800. Step 802 accepts information packets having a variable number of cells, at a plurality of switch inputs, addressed to a plurality of switch outputs. Typically, a plurality of available switch inputs accept information packets addressed to a common switch output. Step 804 queues the information packets into a first plurality of virtual output queues, differentiated by the switch outputs to which the information packets are addressed. Step 806 parses the information packets into lengths of a cell. Step 808 simultaneously nominates switch inputs and arbitrates between a plurality of available switch inputs, where each available switch input has a virtual output queue with an information packet addressing a common switch output. The simultaneous nominations can occur in a plurality of arbitration cycles. Step 808a, for each switch output, nominates the highest priority available switch input from an available input priority list in a first arbitration cycle. Step 808b nominates successively lower priority available switch inputs in subsequent arbitration cycles if the nominated switch input is not selected.

Step 810 selects the least recently used switch input in response to the available input priority list. Step 812 links switch inputs to switch outputs. Step 814 locks the links to transfer information packets in units of a cell per decision cycle, until the transfer is complete.

Step 808a1 arbitrates between the nominating switch outputs for each switch input receiving multiple nominations. Arbitrating between the nominating switch outputs in Step 808a1 includes accepting the least recently used nominating switch outputs.

Accepting the least recently used nominating switch outputs in Step 808a1 includes sub-steps. Step 808a1a establishes a nominating output priority list for each switch input. Step 808a1b accepts nominating switch outputs in response to the nominating output priority list.

In some aspects, arbitrating between the nominating switch outputs in Step 808a1 includes each arbitrating switch input simultaneously accepting a nominating switch output. Arbitrating between the nominating switch outputs in Step 808a1 includes accepting the highest priority nominating switch output every decision cycle.

Fig. 9 is a flowchart illustrating additional details of the method of Fig. 8. The method starts with Fig. 900. Fig. 902 accepts information packets having a variable number of cells, at a plurality of switch inputs, addressed to a plurality of switch outputs. Typically, a plurality of available switch inputs accepts information packets addressed to a common switch output. Step 904 queues the information packets into a first plurality of virtual output queues, differentiated by the switch outputs to which the information packets are addressed. Step 906 parses the information packets into lengths of a cell. Step 908 simultaneously nominates switch inputs and arbitrates between a plurality of available switch inputs, where the available switch inputs each have a virtual output queue with an information packet addressing a common switch output. The simultaneous nomination can occur in a plurality of arbitration cycles. Step 908a creates a sequential input pointer for each available input priority list. Step 908b nominates the highest priority available switch input from the available input priority list in a first arbitration cycle for each switch output. Step 908c, for each switch input receiving multiple nominations, arbitrates between the nominating switch outputs by accepting the least recently used nominating switch output.

Step 908c1 establishes a nominating output priority list for each switch input. Step 908c2 simultaneously accepts nominating switch outputs in response to the nominating output priority list. Step 908c3, following the acceptance of a nominating output by a first switch input, sets the input pointer to a second switch input, next in sequence to the first switch input. Step 908d nominates successively lower priority available switch inputs in subsequent arbitration cycles, if the nominated switch input is not selected. Step 910 selects the switch input. Step 912 links switch inputs to switch outputs. Step 914 locks the links to transfer the information packets in units of a cell per decision cycle, until the transfer is complete. Step 916 nominates the available switch input closest in succession to the second switch input in a subsequent arbitration decision.

Step 908c2a creates a sequential output pointer for each nominating output priority list. Step 908c2b, following the acceptance of a nominating first switch output in a first arbitration cycle, sets the pointer to a second switch output next in sequence to the first switch output. Step 908c2c accepts the nominating switch output closest in succession to second switch output in a subsequent decision cycle.

Step 901 establishes a plurality of arbitration cycles in each decision cycle. Locking the links to transfer the information packets in Step 914 includes transferring one cell per link, every decision cycle.

In some aspects of the invention, arbitrating between the plurality of available switch inputs in Step 908 includes sub-steps. Step 908a sets a maximum number of arbitration cycles in each decision cycle. Step 908b, continues the arbitration cycles if each switch output has not been linked to a switch input. Step 908c ends arbitration if each switch output has been inked to a switch input. Step 908d ends arbitration if the maximum number of arbitration cycles is reached.

Step 915a, for each switch output locked to a switch input, checks the number of cells in the information packet remaining to be transferred. Step 915b bypasses the arbitration process if the number of remaining cells is one or greater. Step 915c arbitrates between the plurality of switch inputs in a subsequent decision cycle if no cells remain in the information packet to be transferred.

Fig. 10 is a flowchart depicting a method for transferring information across a switch connection. The method begins with Step 1000. Step 1002 accepts variable length information packets at a plurality of switch inputs, addressed to a plurality of switch outputs. Step 1004 selects a switch input for each switch output. Step 1006 completely transfers the information packet at the selected switch input to the switch output. Step 1008 reselects a switch input following the completed transfer.

Fig. 11 is a flowchart presenting the arbitration process of Figs. 6 through 10 in a different light. The process detail begins with Step 1000, the step of parsing, equivalent to Step 806 of Fig. 8. In Step 1102 a least recently used (LRU), as defined herein, switch input is nominated. In Step 1104 the available input priority list counter is advanced. In Step 1106 a decision is made as to whether the same switch input has been nominated by the multiple switch outputs. If so, a decision is made in Step 1108 as to whether the maximum number of arbitration cycles have occurred. If not, Step 1110 accepts the least recently used nominating switch output and Step 1112 advances the nominating output priority list pointer. In Step 1114 a decision is made as to whether the accepting switch input is selected. If not, the process returns to Step 1102 and the next least recently used available switch input is nominated. If the accepting switch input is selected, the process continues to Step 1116 where the switch input and switch output links are locked.

A system and method of locking a link to transfer information packets across a switch has been presented above. Examples have been given using a switch with a single crossbar. However, the present invention is equally applicable and scalable to a switch having a plurality of crossbars. For simplification, examples have also been given using a limited number of switch inputs and switch outputs. However, once again, the present invention is not limited to any particular number of interfaces. Further, the invention is not limited to any particular cell size or arbitration cycles per decision cycle. The instant invention is well suited for a synchronous optical network (SONET) packet/cell switch, but is also an application to a broadband communications circuit switch system. Other variations and embodiments will occur to those skilled in the art.

WE CLAIM: